Anypoint Platform: API Design

Summary

This instructor-led course is for API designers, developers, and architects who want to get hands-on experience creating well-designed, modular API definitions using RAML 1.0 and Anypoint Platform. It includes a voucher code to take the *MuleSoft Certified Developer - API Design Associate* exam.

Duration

2 days in-person or online

Objectives

At the end of this course, students should be able to:

- Translate design requirements into API resources and methods.
- Use API designer to create API definitions.
- Use RAML to define API resources, methods, parameters, and responses.
- Document and test APIs.
- Minimize repetition in API definitions using resource types and traits.
- Model data in APIs using data types.
- Modularize APIs using libraries, overlays, and extensions.
- Specify API security schemes.

Prerequisites

There are no course prerequisites.

Setup requirements

- A computer with a minimum screen resolution of 1024x768
- Unrestricted internet access to port 80 (with > 5Mbps download and > 2Mbps upload)
- An Anypoint Platform account
  http://anypoint.mulesoft.com
- The latest version of Firefox or Chrome or Internet Explorer 10 or newer

A detailed setup document can be downloaded from here: https://training.mulesoft.com/downloads

Outline

PART 1: Designing APIs

Module 1: Introducing RESTful API Design

- Describe the common web API formats including SOAP, RPC, and REST
- Describe REST API architecture
- List the rules for retaining REST principles in APIs
- Describe design-first approach for REST APIs

Module 2: Translating Functional Requirements for APIs

- Identify different categories and actions for a REST API
- Convert categories to resources
- Select HTTP methods to support the actions on the categories

Module 3: Introducing API-Led Connectivity and the API Lifecycle

- Describe the API development lifecycle
- Explain MuleSoft's API-led connectivity approach
- Navigate Anypoint Platform
- Describe the API design lifecycle with Anypoint Platform

PART 2: Defining APIs with the RESTful API Modeling Language (RAML)

Module 4: Defining API Resources and Methods

- Use RAML 1.0 to create API definitions
- Define resources and methods in RAML API definitions
- Specify URI parameters for necessary resource methods

Module 5: Specifying Responses

- Describe response structure in HTTP methods
- Use status codes in HTTP responses
- Add error handling and caching information to HTTP responses
- Select and specify the types of content returned in HTTP responses

Module 6: Documenting and Testing APIs

- Add documentation and description nodes to RAML definitions
- Use the mocking service to create API endpoints
- Use the API Console to test API endpoints

Module 7: Making APIs Discoverable

- Create API Portals for learning about and testing APIs
- Customize API Portals with themes
- Publish API definitions to the Anypoint Exchange for discovery
- Gather feedback from API consumers

Module 8: Modeling Data

- Create datatypes and their properties for resources
- Create examples for datatypes
- Include datatypes and examples in resource methods
- Create scenarios in API Notebook to manipulate data using datatypes and examples

Module 9: Reusing Patterns

- Create and reference resource types patterns for reusability
- Use traits to modularize methods

Module 10: Modularizing APIs

- Use libraries for greater API composability
- Use overlays to internationalize resources
- Use extensions to promote portability to test APIs in multiple environments

Module 11: Securing APIs

- Define API security requirements
- Use security schemes to apply resource and method level policies
- Define custom security schemes for APIs
- Apply an OAuth2.0 external provider policy to resource methods

Module 12: Enhancing API Responses using Hypermedia

- Describe hypermedia
- Simplify API discoverability using hypermedia
- Use hypermedia to enhance API responses
- Modify API definitions to generate state-specific client responses in resource methods

Module 13: Versioning APIs

- Explain when and when not to version APIs
- Describe the methods for versioning APIs
- Document changes in new API versions using shared API Portals
- Deprecate older versions of APIs